# 使用逻辑应用针对新增或已更改的 DocumentDB 资源发送通知

这篇文章的灵感来自我在某一个 DocumentDB 社区论坛上看到的一个问题。这个问题是 **DocumentDB 是否支持针对已修改的资源发送通知**？

我已使用 BizTalk 服务器许多年，这是使用 [WCF LOB 适配器](https://msdn.microsoft.com/library/bb798128.aspx)时非常常见的案例。因此，我决定试试看能否在 DocumentDB 中对新增和/或已修改的文档重现此功能。

本文概述了更改通知解决方案的组件，其中包括[触发器](/documentation/articles/documentdb-programming#trigger)和[逻辑应用](/documentation/articles/app-service-logic-what-are-logic-apps)。重要代码段以内联方式提供，你可以在 [GitHub](https://github.com/HEDIDIN/DocDbNotifications) 上获取整个解决方案。

## 使用案例

下列案例是本文的使用案例。

DocumentDB 是 Health Level Seven International (HL7) Fast Healthcare Interoperability Resources (FHIR) 文档的存储库。假设你的 DocumentDB 数据库与你的 API 和逻辑应用一起构成 HL7 FHIR Server。医疗保健机构会将患者的数据存储在 DocumentDB 的“Patients”数据库中。患者数据库中有数个集合：Clinical、Identification 等等。患者信息属于身份信息。你有一个名为“Patient”的集合。

心脏病科会跟踪个人健康和锻炼数据。搜索新增或已修改的患者记录相当耗时。他们询问 IT 部门是否有办法让他们收到新增或已修改患者记录的通知。

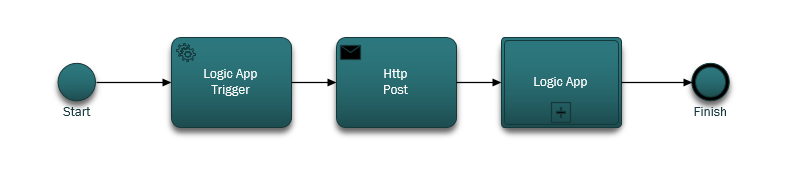
IT 部门表示他们可以轻松提供此通知。他们还表示可以将文档推送到 [Azure Blob 存储](/services/storage/)，以便心脏病科轻松访问。

## IT 部门如何解决此问题

为了创建此应用程序，IT 部门决定先创建其模型。使用业务流程模型和标注 (BPMN) 的好处就是技术和非技术人员都可以轻松了解。这整个通知流程会被视为业务流程。

## 通知流程的高级视图

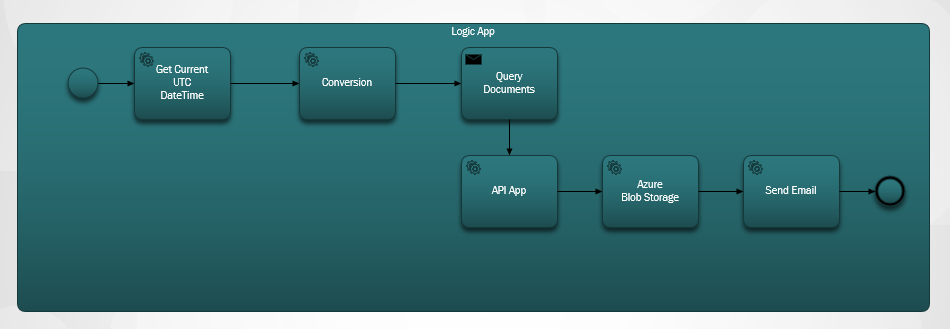
1. 从具有计时器触发器的逻辑应用开始。默认情况下，每小时都会运行触发器。
2. 接下来，对逻辑应用执行 HTTP POST。
3. 逻辑应用会执行所有任务。



高级视图

### 让我们看一下此逻辑应用的用途

如果查看下图，你会发现 LogicApp 工作流中有几个步骤。



主要逻辑流程

步骤如下：

1. 你需要从 API 应用获取当前的 UTC 日期时间。默认值为一小时前。
2. 将 UTC 日期时间转换成 Unix 时间戳格式。这是 DocumentDB 中时间戳的默认格式。
3. 将此值 POST 到 API 应用，这会进行 DocumentDB 查询。此值用于查询中。

* SELECT \* FROM Patients p WHERE (p.\_ts >= @unixTimeStamp)
* [AZURE.NOTE] \_ts 表示所有 DocumentDB 资源的时间戳元数据。

1. 如果找到文档，则会将响应正文发送到 Azure Blob 存储。

* [AZURE.NOTE] Blob 存储需要 Azure 存储帐户。你必须预配 Azure Blob 存储帐户，并添加名为 patients 的新 Blob。有关详细信息，请参阅[关于 Azure 存储帐户](../storage/storage-create-storage-account.md)和[开始使用 Azure Blob 存储](../storage/storage-dotnet-how-to-use-blobs.md)。

1. 最后会发送电子邮件，通知收件人已找到的文档数目。如果找不到任何文档，电子邮件正文将为“0 Documents Found”。

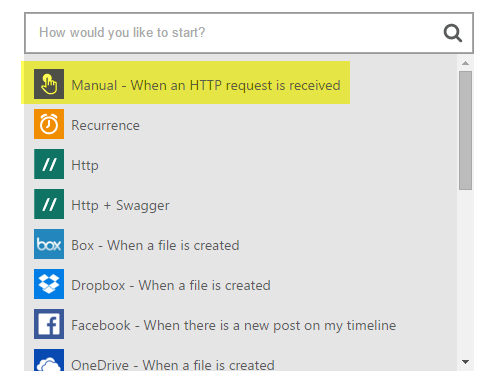
你现在已了解该工作流的用途，让我们看看其实施方式。

### 从主要逻辑应用开始

如果你不熟悉逻辑应用，可以从 [Azure 应用商店](https://portal.azure.cn/)中获取，而且你可以在[什么是逻辑应用？](/documentation/articles/app-service-logic-what-are-logic-apps)中进一步了解

创建新的逻辑应用时，系统会询问你**你要如何开始？**

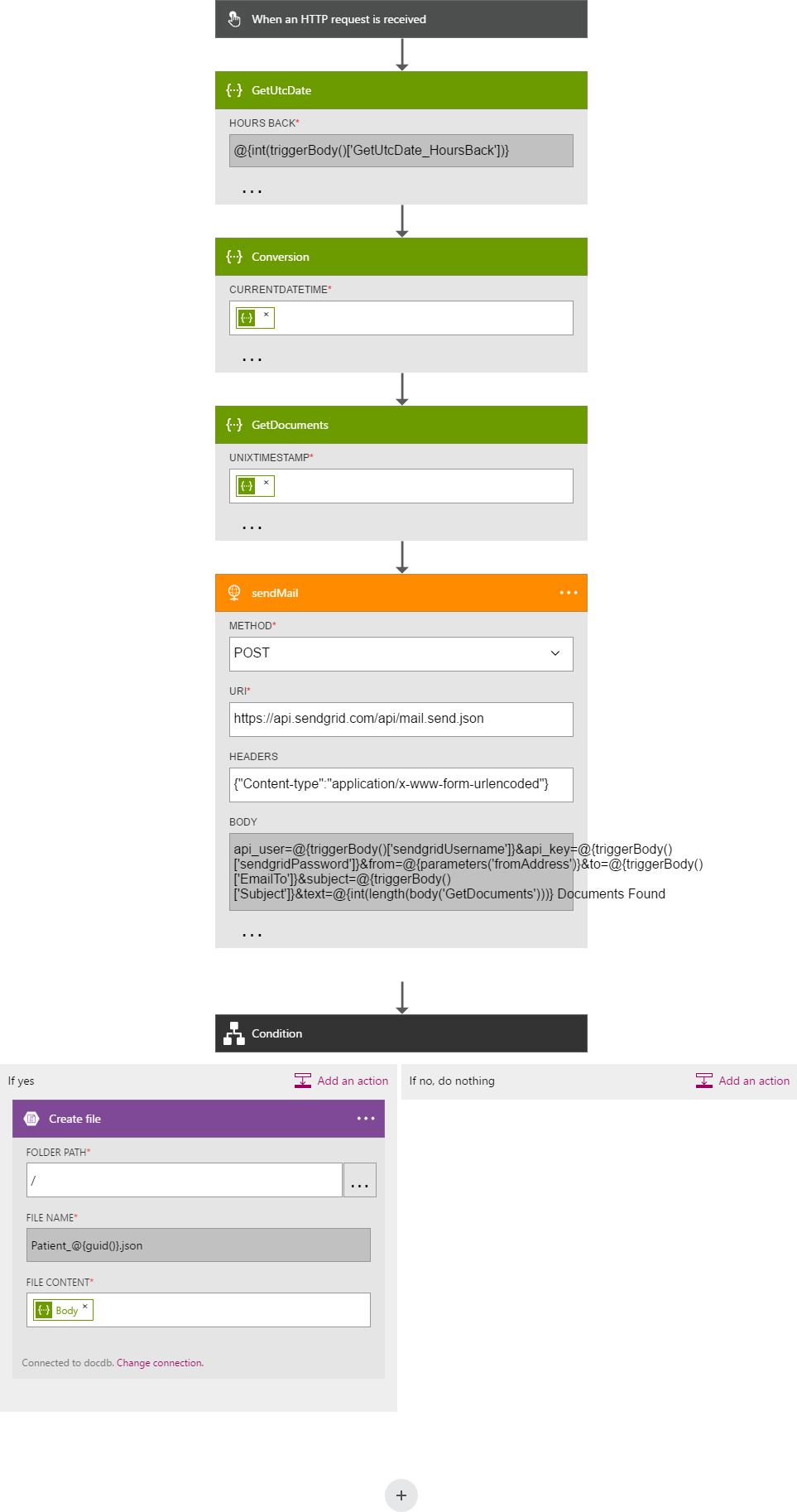
当你单击文本框内部时，有各种事件可供选择。对于此逻辑应用，选择“手动 - 收到 HTTP 请求时”，如下所示。



开始进行

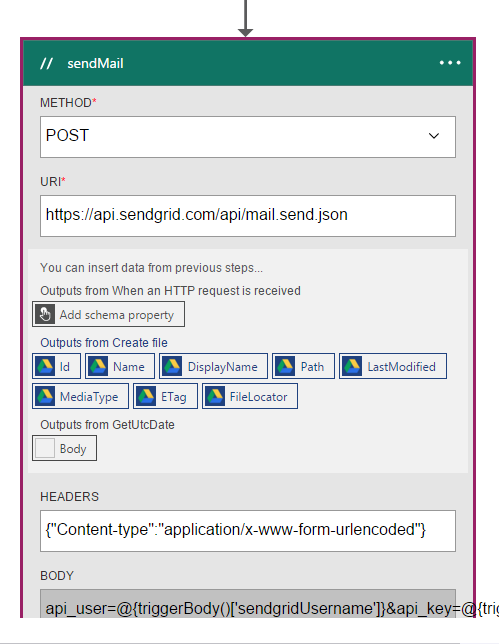
### 完整逻辑应用的设计视图

让我们往前跳并查看逻辑应用（名为 DocDB）的完整设计视图。



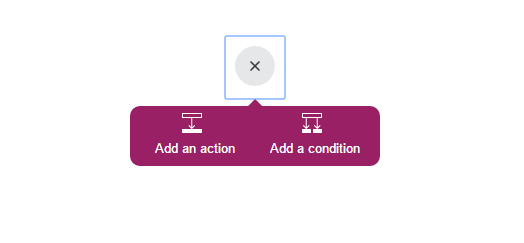
逻辑应用工作流

在逻辑应用设计器中编辑操作时，你可以选择来自 HTTP请求或来自前一个操作的**输出**，如以下 sendMail 操作所示。



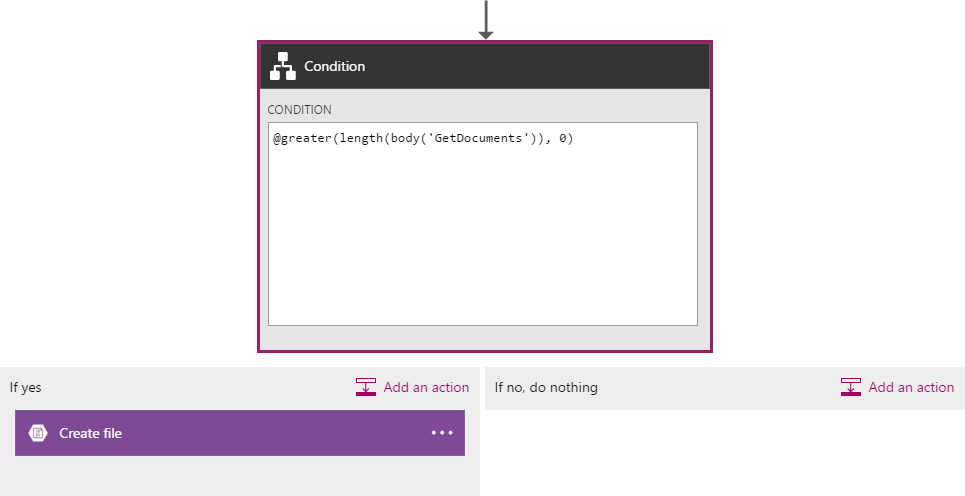
选择输出

在执行工作流中的每个操作之前，你可以做出决定；如下图所示“添加操作”或“添加条件”。



做出决定

如果你选择“添加条件”，即会出现如下图所示的窗体，以便输入你的逻辑。这其实就是业务规则。如果单击字段内部，可以选择来自前一个操作的参数。也可以直接输入值。



添加条件

[AZURE.NOTE] 你也可以在“代码视图”中输入任何信息。

让我们在代码视图中看一下完整的逻辑应用。

"$schema": "https://schema.management.azure.com/providers/Microsoft.Logic/schemas/2015-08-01-preview/workflowdefinition.json#",  
 "actions": {  
 "Conversion": {  
 "conditions": [  
 {  
 "dependsOn": "GetUtcDate"  
 }  
 ],  
 "inputs": {  
 "method": "post",  
 "queries": {  
 "currentdateTime": "@{body('GetUtcDate')}"  
 },  
 "uri": "https://docdbnotificationapi-debug.azurewebsites.net/api/Conversion"  
 },  
 "metadata": {  
 "apiDefinitionUrl": "https://docdbnotificationapi-debug.azurewebsites.net/swagger/docs/v1",  
 "swaggerSource": "custom"  
 },  
 "type": "Http"  
 },  
 "Createfile": {  
 "conditions": [  
 {  
 "expression": "@greater(length(body('GetDocuments')), 0)"  
 },  
 {  
 "dependsOn": "GetDocuments"  
 }  
 ],  
 "inputs": {  
 "body": "@body('GetDocuments')",  
 "host": {  
 "api": {  
 "runtimeUrl": "https://logic-apis-westus.azure-apim.net/apim/azureblob"  
 },  
 "connection": {  
 "name": "@parameters('$connections')['azureblob']['connectionId']"  
 }  
 },  
 "method": "post",  
 "path": "/datasets/default/files",  
 "queries": {  
 "folderPath": "/patients",  
 "name": "Patient\_@{guid()}.json"  
 }  
 },  
 "type": "ApiConnection"  
 },  
 "GetDocuments": {  
 "conditions": [  
 {  
 "dependsOn": "Conversion"  
 }  
 ],  
 "inputs": {  
 "method": "post",  
 "queries": {  
 "unixTimeStamp": "@body('Conversion')"  
 },  
 "uri": "https://docdbnotificationapi-debug.azurewebsites.net/api/Patient"  
 },  
 "metadata": {  
 "apiDefinitionUrl": "https://docdbnotificationapi-debug.azurewebsites.net/swagger/docs/v1",  
 "swaggerSource": "custom"  
 },  
 "type": "Http"  
 },  
 "GetUtcDate": {  
 "conditions": [],  
 "inputs": {  
 "method": "get",  
 "queries": {  
 "hoursBack": "@{int(triggerBody()['GetUtcDate\_HoursBack'])}"  
 },  
 "uri": "https://docdbnotificationapi-debug.azurewebsites.net/api/Authorization"  
 },  
 "metadata": {  
 "apiDefinitionUrl": "https://docdbnotificationapi-debug.azurewebsites.net/swagger/docs/v1",  
 "swaggerSource": "custom"  
 },  
 "type": "Http"  
 },  
 "sendMail": {  
 "conditions": [  
 {  
 "dependsOn": "GetDocuments"  
 }  
 ],  
 "inputs": {  
 "body": "api\_user=@{triggerBody()['sendgridUsername']}&api\_key=@{triggerBody()['sendgridPassword']}&from=@{parameters('fromAddress')}&to=@{triggerBody()['EmailTo']}&subject=@{triggerBody()['Subject']}&text=@{int(length(body('GetDocuments')))} Documents Found",  
 "headers": {  
 "Content-type": "application/x-www-form-urlencoded"  
 },  
 "method": "POST",  
 "uri": "https://api.sendgrid.com/api/mail.send.json"  
 },  
 "type": "Http"  
 }  
 },  
 "contentVersion": "1.0.0.0",  
 "outputs": {  
 "Results": {  
 "type": "String",  
 "value": "@{int(length(body('GetDocuments')))} Records Found"  
 }  
 },  
 "parameters": {  
 "$connections": {  
 "defaultValue": {},  
 "type": "Object"  
 },  
 "fromAddress": {  
 "defaultValue": "user@msn.com",  
 "type": "String"  
 },  
 "toAddress": {  
 "defaultValue": "XXXXX@XXXXXXX.net",  
 "type": "String"  
 }  
 },  
 "triggers": {  
 "manual": {  
 "inputs": {  
 "schema": {  
 "properties": {},  
 "required": [],  
 "type": "object"  
 }  
 },  
 "type": "Manual"  
 }

如果你不熟悉代码中各节所代表的意义，可以查看[逻辑应用工作流定义语言](http://aka.ms/logicappsdocs)文档。

在此工作流中，你会使用 [HTTP Webhook 触发器](https://sendgrid.com/blog/whats-webhook/)。如果查看上述代码，你会看到以下示例所示的参数。

=@{triggerBody()['Subject']}

triggerBody() 代表逻辑应用 REST API 的 REST POST 主体中包含的参数。()['Subject'] 代表字段。所有这些参数构成了 JSON 格式的主体。

[AZURE.NOTE] 使用 Webhook，你可以完整访问触发器的请求标头和主体。在此应用程序中，你会需要主体。

如先前所述，你可以使用设计器来分配参数，或在代码视图中分配参数。 如果在代码视图中分配参数，你会接着定义需有值的属性，如下列代码示例所示。

"triggers": {  
 "manual": {  
 "inputs": {  
 "schema": {  
 "properties": {  
 "Subject": {  
 "type" : "String"   
  
 }  
 },  
 "required": [  
 "Subject"  
 ],  
 "type": "object"  
 }  
 },  
 "type": "Manual"  
 }  
 }

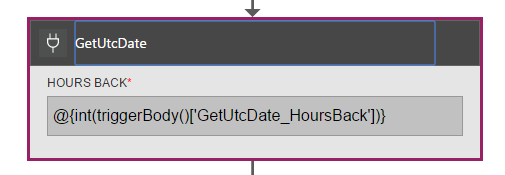
你正在创建将从 HTTP POST 主体传入的 JSON 架构。 若要引发触发器，你需要一个回叫 URL。你将在稍后的教程中了解如何生成回叫 URL。

## 操作

我们来看一下逻辑应用中每个操作的用途。

### GetUTCDate

**设计器视图**



**代码视图**

"GetUtcDate": {  
 "conditions": [],  
 "inputs": {  
 "method": "get",  
 "queries": {  
 "hoursBack": "@{int(triggerBody()['GetUtcDate\_HoursBack'])}"  
 },  
 "uri": "https://docdbnotificationapi-debug.azurewebsites.net/api/Authorization"  
 },  
 "metadata": {  
 "apiDefinitionUrl": "https://docdbnotificationapi-debug.azurewebsites.net/swagger/docs/v1"  
 },  
 "type": "Http"  
 },

此 HTTP 操作会执行 GET 操作。它会调用 API 应用 GetUtcDate 方法。Uri 使用传入触发器主体的“GetUtcDate\_HoursBack”属性。“GetUtcDate\_HoursBack”值在第一个逻辑应用中设置。你将在稍后的教程中详细了解触发器逻辑应用。

此操作会调用 API 应用以返回 UTC 日期字符串值。

#### 操作

**请求**

{  
 "uri": "https://docdbnotificationapi-debug.azurewebsites.net/api/Authorization",  
 "method": "get",  
 "queries": {  
 "hoursBack": "24"  
 }  
 }

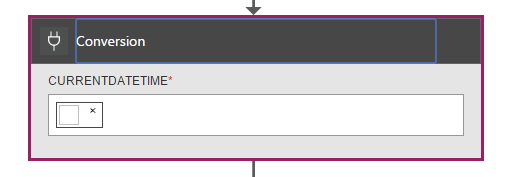
**响应**

{  
 "statusCode": 200,  
 "headers": {  
 "pragma": "no-cache",  
 "cache-Control": "no-cache",  
 "date": "Fri, 26 Feb 2016 15:47:33 GMT",  
 "server": "Microsoft-IIS/8.0",  
 "x-AspNet-Version": "4.0.30319",  
 "x-Powered-By": "ASP.NET"  
 },  
 "body": "Fri, 15 Jan 2016 23:47:33 GMT"  
 }

下一步是将 UTC 日期时间值转换为 Unix 时间戳，后者是 .NET double 类型。

### 转换

##### 设计器视图



转换

##### 代码视图

"Conversion": {  
 "conditions": [  
 {  
 "dependsOn": "GetUtcDate"  
 }  
 ],  
 "inputs": {  
 "method": "post",  
 "queries": {  
 "currentDateTime": "@{body('GetUtcDate')}"  
 },  
 "uri": "https://docdbnotificationapi-debug.azurewebsites.net/api/Conversion"  
 },  
 "metadata": {  
 "apiDefinitionUrl": "https://docdbnotificationapi-debug.azurewebsites.net/swagger/docs/v1"  
 },  
 "type": "Http"  
 },

在此步骤中，你会传入从 GetUTCDate 返回的值。系统有一个 dependsOn 条件，这表示必须成功完成 GetUTCDate 操作。如果未成功完成，则跳过此操作。

此操作会调用 API 应用以处理转换。

#### 操作

##### 请求

{  
 "uri": "https://docdbnotificationapi-debug.azurewebsites.net/api/Conversion",  
 "method": "post",  
 "queries": {  
 "currentDateTime": "Fri, 15 Jan 2016 23:47:33 GMT"  
 }  
 }

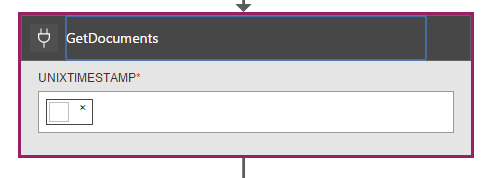
##### 响应

{  
 "statusCode": 200,  
 "headers": {  
 "pragma": "no-cache",  
 "cache-Control": "no-cache",  
 "date": "Fri, 26 Feb 2016 15:47:33 GMT",  
 "server": "Microsoft-IIS/8.0",  
 "x-AspNet-Version": "4.0.30319",  
 "x-Powered-By": "ASP.NET"  
 },  
 "body": 1452901653  
 }

在下一个操作中，你将对我们的 API 应用执行 POST 操作。

### GetDocuments

##### 设计器视图



获取文档

##### 代码视图

"GetDocuments": {  
 "conditions": [  
 {  
 "dependsOn": "Conversion"  
 }  
 ],  
 "inputs": {  
 "method": "post",  
 "queries": {  
 "unixTimeStamp": "@{body('Conversion')}"  
 },  
 "uri": "https://docdbnotificationapi-debug.azurewebsites.net/api/Patient"  
 },  
 "metadata": {  
 "apiDefinitionUrl": "https://docdbnotificationapi-debug.azurewebsites.net/swagger/docs/v1"  
 },  
 "type": "Http"  
 },

在 GetDocuments 操作中，你将传入来自 Conversion 操作的响应主体。这是 Uri 中的参数：

unixTimeStamp=@{body('Conversion')}

QueryDocuments 操作会对 API 应用执行 HTTP POST 操作。

调用的方法为 **QueryForNewPatientDocuments**。

#### 操作

##### 请求

{  
 "uri": "https://docdbnotificationapi-debug.azurewebsites.net/api/Patient",  
 "method": "post",  
 "queries": {  
 "unixTimeStamp": "1452901653"  
 }  
 }

##### 响应

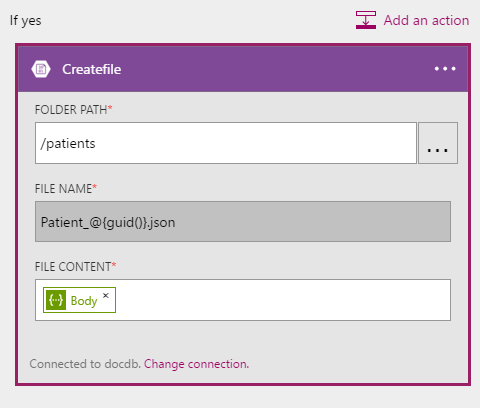
{  
 "statusCode": 200,  
 "headers": {  
 "pragma": "no-cache",  
 "cache-Control": "no-cache",  
 "date": "Fri, 26 Feb 2016 15:47:35 GMT",  
 "server": "Microsoft-IIS/8.0",  
 "x-AspNet-Version": "4.0.30319",  
 "x-Powered-By": "ASP.NET"  
 },  
 "body": [  
 {  
 "id": "xcda",  
 "\_rid": "vCYLAP2k6gAXAAAAAAAAAA==",  
 "\_self": "dbs/vCYLAA==/colls/vCYLAP2k6gA=/docs/vCYLAP2k6gAXAAAAAAAAAA==/",  
 "\_ts": 1454874620,  
 "\_etag": "\"00007d01-0000-0000-0000-56b79ffc0000\"",  
 "resourceType": "Patient",  
 "text": {  
 "status": "generated",  
 "div": "<div>\n \n <p>Henry Levin the 7th</p>\n \n </div>"  
 },  
 "identifier": [  
 {  
 "use": "usual",  
 "type": {  
 "coding": [  
 {  
 "system": "http://hl7.org/fhir/v2/0203",  
 "code": "MR"  
 }  
 ]  
 },  
 "system": "urn:oid:2.16.840.1.113883.19.5",  
 "value": "12345"  
 }  
 ],  
 "active": true,  
 "name": [  
 {  
 "family": [  
 "Levin"  
 ],  
 "given": [  
 "Henry"  
 ]  
 }  
 ],  
 "gender": "male",  
 "birthDate": "1932-09-24",  
 "managingOrganization": {  
 "reference": "Organization/2.16.840.1.113883.19.5",  
 "display": "Good Health Clinic"  
 }  
 },

下一个操作是将文档保存到 [Azure Blog 存储](https://azure.microsoft.com/services/storage/)。

[AZURE.NOTE] Blob 存储需要 Azure 存储帐户。你必须预配 Azure Blob 存储帐户，并添加名为 patients 的新 Blob。有关详细信息，请参阅[开始使用 Azure Blob 存储](../storage/storage-dotnet-how-to-use-blobs.md)。

### 创建文件

##### 设计器视图



创建文件

##### 代码视图

{  
 "host": {  
 "api": {  
 "runtimeUrl": "https://logic-apis-westus.azure-apim.net/apim/azureblob"  
 },  
 "connection": {  
 "name": "subscriptions/fxxxxxc079-4e5d-b002-xxxxxxxxxx/resourceGroups/Api-Default-Central-US/providers/Microsoft.Web/connections/azureblob"  
 }  
 },  
 "method": "post",  
 "path": "/datasets/default/files",  
 "queries": {  
 "folderPath": "/patients",  
 "name": "Patient\_17513174-e61d-4b56-88cb-5cf383db4430.json"  
 },  
 "body": [  
 {  
 "id": "xcda",  
 "\_rid": "vCYLAP2k6gAXAAAAAAAAAA==",  
 "\_self": "dbs/vCYLAA==/colls/vCYLAP2k6gA=/docs/vCYLAP2k6gAXAAAAAAAAAA==/",  
 "\_ts": 1454874620,  
 "\_etag": "\"00007d01-0000-0000-0000-56b79ffc0000\"",  
 "resourceType": "Patient",  
 "text": {  
 "status": "generated",  
 "div": "<div>\n \n <p>Henry Levin the 7th</p>\n \n </div>"  
 },  
 "identifier": [  
 {  
 "use": "usual",  
 "type": {  
 "coding": [  
 {  
 "system": "http://hl7.org/fhir/v2/0203",  
 "code": "MR"  
 }  
 ]  
 },  
 "system": "urn:oid:2.16.840.1.113883.19.5",  
 "value": "12345"  
 }  
 ],  
 "active": true,  
 "name": [  
 {  
 "family": [  
 "Levin"  
 ],  
 "given": [  
 "Henry"  
 ]  
 }  
 ],  
 "gender": "male",  
 "birthDate": "1932-09-24",  
 "managingOrganization": {  
 "reference": "Organization/2.16.840.1.113883.19.5",  
 "display": "Good Health Clinic"  
 }  
 },

此代码通过设计器中的操作生成。你不需要修改此代码。

如果不熟悉如何使用 Azure Blob API，请参阅[开始使用 Azure Blob 存储 API](../connectors/connectors-create-api-azureblobstorage.md)。

#### 操作

##### 请求

"host": {  
 "api": {  
 "runtimeUrl": "https://logic-apis-westus.azure-apim.net/apim/azureblob"  
 },  
 "connection": {  
 "name": "subscriptions/fxxxxxc079-4e5d-b002-xxxxxxxxxx/resourceGroups/Api-Default-Central-US/providers/Microsoft.Web/connections/azureblob"  
 }  
 },  
 "method": "post",  
 "path": "/datasets/default/files",  
 "queries": {  
 "folderPath": "/patients",  
 "name": "Patient\_17513174-e61d-4b56-88cb-5cf383db4430.json"  
 },  
 "body": [  
 {  
 "id": "xcda",  
 "\_rid": "vCYLAP2k6gAXAAAAAAAAAA==",  
 "\_self": "dbs/vCYLAA==/colls/vCYLAP2k6gA=/docs/vCYLAP2k6gAXAAAAAAAAAA==/",  
 "\_ts": 1454874620,  
 "\_etag": "\"00007d01-0000-0000-0000-56b79ffc0000\"",  
 "resourceType": "Patient",  
 "text": {  
 "status": "generated",  
 "div": "<div>\n \n <p>Henry Levin the 7th</p>\n \n </div>"  
 },  
 "identifier": [  
 {  
 "use": "usual",  
 "type": {  
 "coding": [  
 {  
 "system": "http://hl7.org/fhir/v2/0203",  
 "code": "MR"  
 }  
 ]  
 },  
 "system": "urn:oid:2.16.840.1.113883.19.5",  
 "value": "12345"  
 }  
 ],  
 "active": true,  
 "name": [  
 {  
 "family": [  
 "Levin"  
 ],  
 "given": [  
 "Henry"  
 ]  
 }  
 ],  
 "gender": "male",  
 "birthDate": "1932-09-24",  
 "managingOrganization": {  
 "reference": "Organization/2.16.840.1.113883.19.5",  
 "display": "Good Health Clinic"  
 }  
 },….

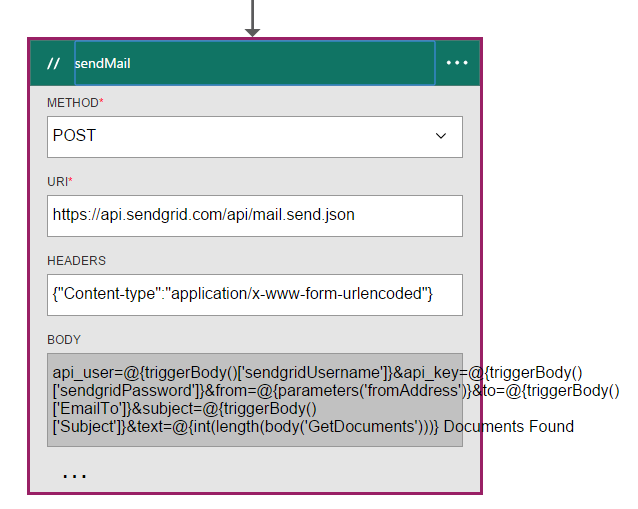
##### 响应

{  
 "statusCode": 200,  
 "headers": {  
 "pragma": "no-cache",  
 "x-ms-request-id": "2b2f7c57-2623-4d71-8e53-45c26b30ea9d",  
 "cache-Control": "no-cache",  
 "date": "Fri, 26 Feb 2016 15:47:36 GMT",  
 "set-Cookie": "ARRAffinity=29e552cea7db23196f7ffa644003eaaf39bc8eb6dd555511f669d13ab7424faf;Path=/;Domain=127.0.0.1",  
 "server": "Microsoft-HTTPAPI/2.0",  
 "x-AspNet-Version": "4.0.30319",  
 "x-Powered-By": "ASP.NET"  
 },  
 "body": {  
 "Id": "0B0nBzHyMV-\_NRGRDcDNMSFAxWFE",  
 "Name": "Patient\_47a2a0dc-640d-4f01-be38-c74690d085cb.json",  
 "DisplayName": "Patient\_47a2a0dc-640d-4f01-be38-c74690d085cb.json",  
 "Path": "/Patient/Patient\_47a2a0dc-640d-4f01-be38-c74690d085cb.json",  
 "LastModified": "2016-02-26T15:47:36.215Z",  
 "Size": 65647,  
 "MediaType": "application/octet-stream",  
 "IsFolder": false,  
 "ETag": "\"c-g\_a-1OtaH-kNQ4WBoXLp3Zv9s/MTQ1NjUwMTY1NjIxNQ\"",  
 "FileLocator": "0B0nBzHyMV-\_NRGRDcDNMSFAxWFE"  
 }  
 }

最后一步是发送电子邮件通知

### sendEmail

##### 设计器视图



发送电子邮件

##### 代码视图

"sendMail": {  
 "conditions": [  
 {  
 "dependsOn": "GetDocuments"  
 }  
 ],  
 "inputs": {  
 "body": "api\_user=@{triggerBody()['sendgridUsername']}&api\_key=@{triggerBody()['sendgridPassword']}&from=@{parameters('fromAddress')}&to=@{triggerBody()['EmailTo']}&subject=@{triggerBody()['Subject']}&text=@{int(length(body('GetDocuments')))} Documents Found",  
 "headers": {  
 "Content-type": "application/x-www-form-urlencoded"  
 },  
 "method": "POST",  
 "uri": "https://api.sendgrid.com/api/mail.send.json"  
 },  
 "type": "Http"  
 }

在此操作中，你会发送电子邮件通知。你会使用 [SendGrid](https://sendgrid.com/marketing/sendgrid-services?cvosrc=PPC.Bing.sendgrib&cvo_cid=SendGrid%20-%20US%20-%20Brand%20-%20&mc=Paid%20Search&mcd=BingAds&keyword=sendgrib&network=o&matchtype=e&mobile=&content=&search=1&utm_source=bing&utm_medium=cpc&utm_term=%5Bsendgrib%5D&utm_content=%21acq%21v2%2134335083397-8303227637-1649139544&utm_campaign=SendGrid+-+US+-+Brand+-+%28English%29)。

其代码是使用逻辑应用的模板以及 [101-logic-app-sendgrid Github 存储库](https://github.com/Azure/azure-quickstart-templates/tree/master/101-logic-app-sendgrid)中的 SendGrid 生成的。

HTTP 操作是一个 POST。

授权参数位于触发器属性中

},  
 "sendgridPassword": {  
 "type": "SecureString"  
 },  
 "sendgridUsername": {  
 "type": "String"  
 }  
  
 In addition, other parameters are static values set in the Parameters section of the Logic App. These are:  
 },  
 "toAddress": {  
 "defaultValue": "XXXX@XXXX.com",  
 "type": "String"  
 },  
 "fromAddress": {  
 "defaultValue": "XXX@msn.com",  
 "type": "String"  
 },  
 "emailBody": {  
 "defaultValue": "@{string(concat(int(length(actions('QueryDocuments').outputs.body)) Records Found),'/n', actions('QueryDocuments').outputs.body)}",  
 "type": "String"  
 },

emailBody 会将查询所返回的文档数目（可能是“0”或更多）与“Records Found”串连在一起。其余参数从触发器参数设置。

此操作取决于 **GetDocuments** 操作。

#### 操作

##### 请求

{  
 "uri": "https://api.sendgrid.com/api/mail.send.json",  
 "method": "POST",  
 "headers": {  
 "Content-type": "application/x-www-form-urlencoded"  
 },  
 "body": "api\_user=azureuser@azure.com&api\_key=Biz@Talk&from=user@msn.com&to=XXXX@XXXX.com&subject=New Patients&text=37 Documents Found"  
 }

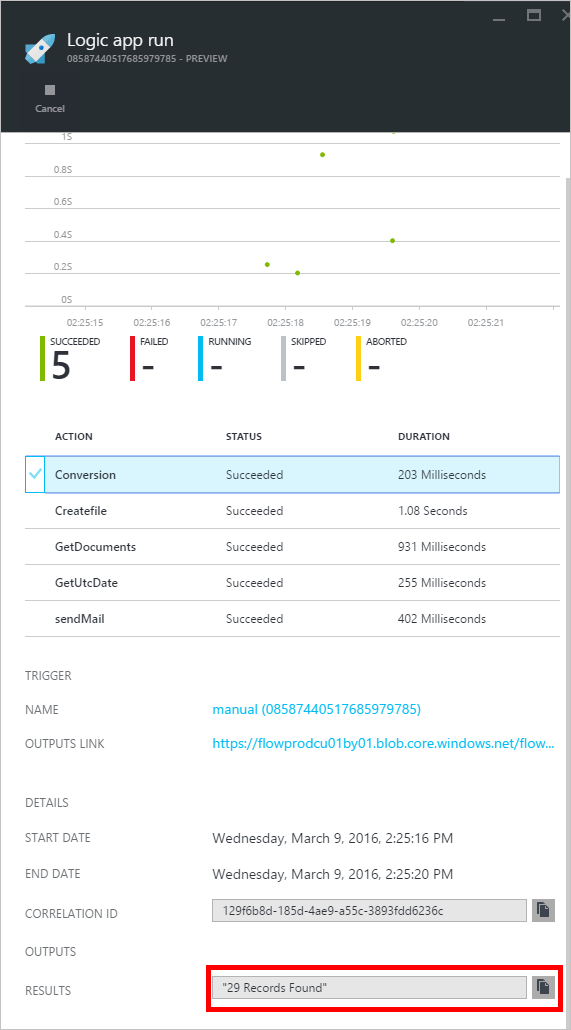
##### 响应

{  
 "statusCode": 200,  
 "headers": {  
 "connection": "keep-alive",  
 "x-Frame-Options": "DENY,DENY",  
 "access-Control-Allow-Origin": "https://sendgrid.com",  
 "date": "Fri, 26 Feb 2016 15:47:35 GMT",  
 "server": "nginx"  
 },  
 "body": {  
 "message": "success"  
 }  
 }

最后，你要能够在 Azure 门户上看到逻辑应用的结果。若要这么做，请向 outputs 节添加参数。

"outputs": {  
 "Results": {  
 "type": "String",  
 "value": "@{int(length(actions('QueryDocuments').outputs.body))} Records Found"  
 }

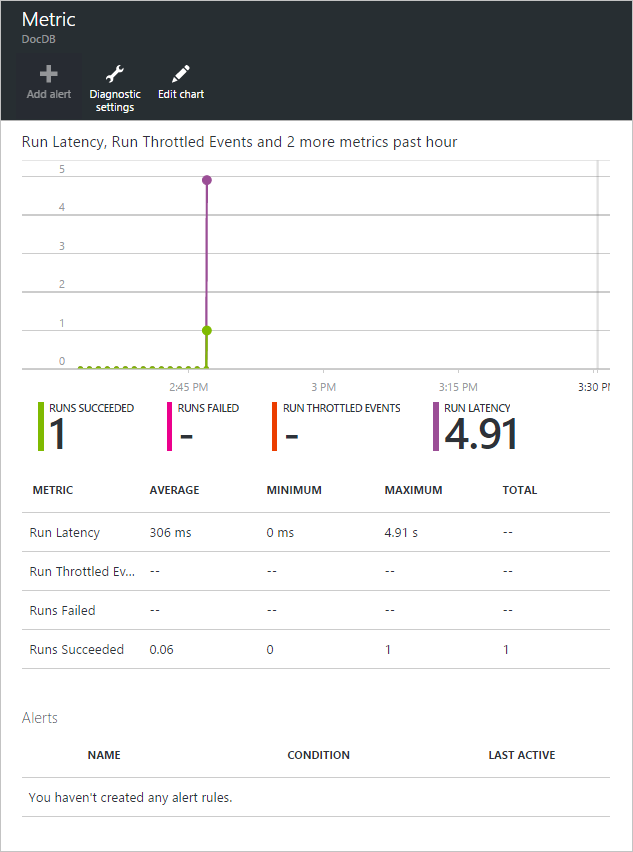
这会返回在电子邮件正文中发送的相同值。下图显示“29 Records Found”的示例。



结果

## 度量值

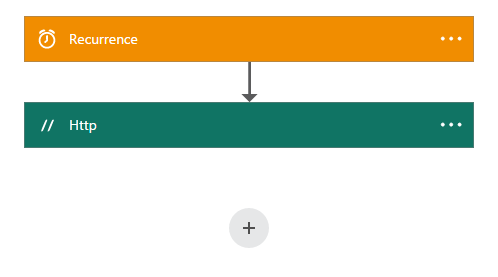
你可以在门户中为主要逻辑应用配置监视。这样，你就可以查看“运行延迟”和其他事件，如下图所示。



## DocDb 触发器

此逻辑应用是在主要逻辑应用上启动工作流的触发器。

下图显示设计器视图。



{  
 "$schema": "https://schema.management.azure.com/providers/Microsoft.Logic/schemas/2015-08-01-preview/workflowdefinition.json#",  
 "actions": {  
 "Http": {  
 "conditions": [],  
 "inputs": {  
 "body": {  
 "EmailTo": "XXXXXX@XXXXX.net",  
 "GetUtcDate\_HoursBack": "24",  
 "Subject": "New Patients",  
 "sendgridPassword": "\*\*\*\*\*\*\*\*",  
 "sendgridUsername": "azureuser@azure.com"  
 },  
 "method": "POST",  
 "uri": "https://prod-01.westus.logic.azure.com:443/workflows/12a1de57e48845bc9ce7a247dfabc887/triggers/manual/run?api-version=2015-08-01-preview&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=ObTlihr529ATIuvuG-dhxOgBL4JZjItrvPQ8PV6973c"  
 },  
 "type": "Http"  
 }  
 },  
 "contentVersion": "1.0.0.0",  
 "outputs": {  
 "Results": {  
 "type": "String",  
 "value": "@{body('Http')['status']}"  
 }  
 },  
 "parameters": {},  
 "triggers": {  
 "recurrence": {  
 "recurrence": {  
 "frequency": "Hour",  
 "interval": 24  
 },  
 "type": "Recurrence"  
 }  
 }  
 }

此触发器已设置为 24 个小时重复一次。 操作为 HTTP POST，该操作使用主要逻辑应用的回叫 URL。 主体包含 JSON 架构中指定的参数。

#### 操作

##### 请求

{  
 "uri": "https://prod-01.westus.logic.azure.com:443/workflows/12a1de57e48845bc9ce7a247dfabc887/triggers/manual/run?api-version=2015-08-01-preview&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=ObTlihr529ATIuvuG-dhxOgBL4JZjItrvPQ8PV6973c",  
 "method": "POST",  
 "body": {  
 "EmailTo": "XXXXXX@XXXXX.net",  
 "GetUtcDate\_HoursBack": "24",  
 "Subject": "New Patients",  
 "sendgridPassword": "\*\*\*\*\*\*\*\*",  
 "sendgridUsername": "azureuser@azure.com"  
 }  
 }

##### 响应

{  
 "statusCode": 202,  
 "headers": {  
 "pragma": "no-cache",  
 "x-ms-ratelimit-remaining-workflow-writes": "7486",  
 "x-ms-ratelimit-burst-remaining-workflow-writes": "1248",  
 "x-ms-request-id": "westus:2d440a39-8ba5-4a9c-92a6-f959b8d2357f",  
 "cache-Control": "no-cache",  
 "date": "Thu, 25 Feb 2016 21:01:06 GMT"  
 }  
 }

现在，让我们看看 API 应用。

## DocDBNotificationApi

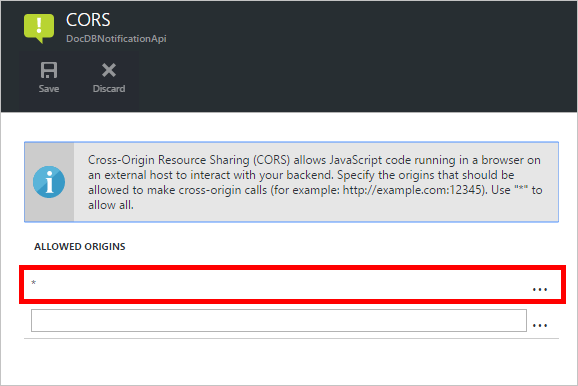
虽然应用中有数个操作，但你只会使用三个。

* GetUtcDate
* ConvertToTimeStamp
* QueryForNewPatientDocuments

### DocDBNotificationApi 操作

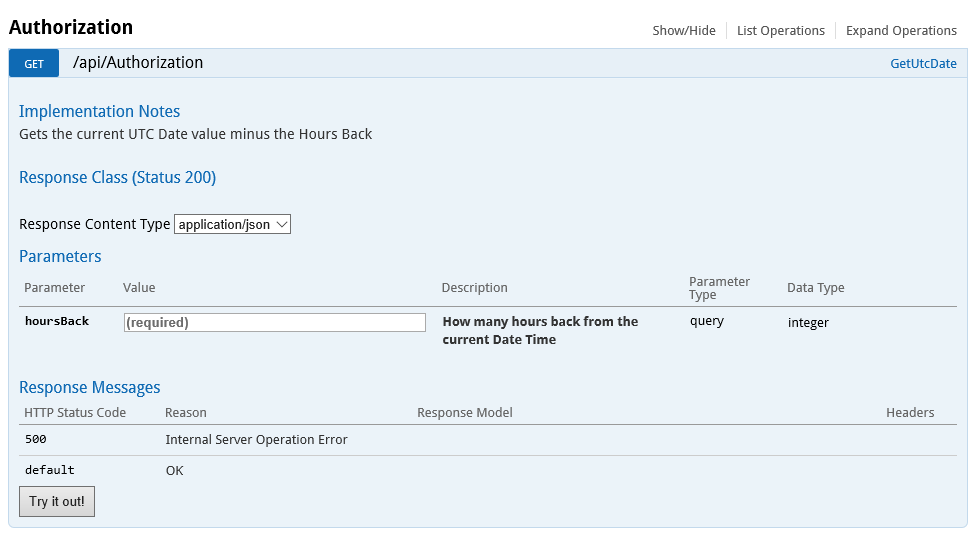
让我们看看 Swagger 文档

[AZURE.NOTE] 为了从外部调用操作，你需要在 API 应用的设置中添加 CORS 允许的原始值“\*”（不含引号），如下图所示。



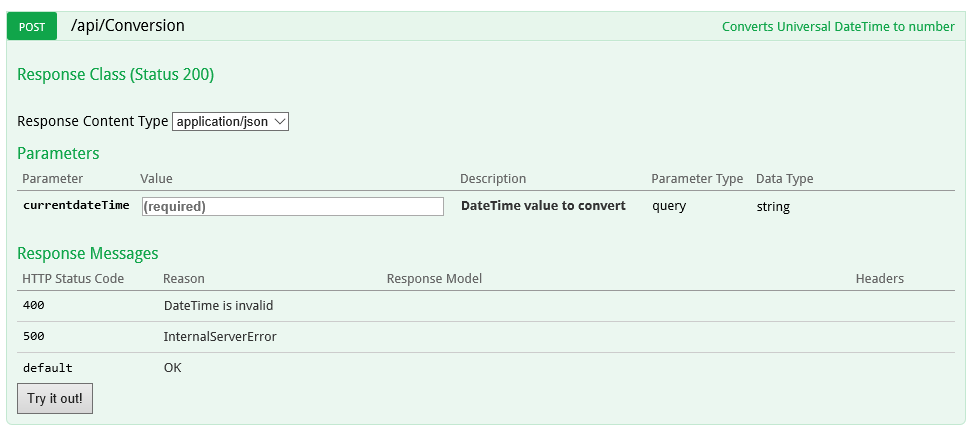
Cors 配置

#### GetUtcDate



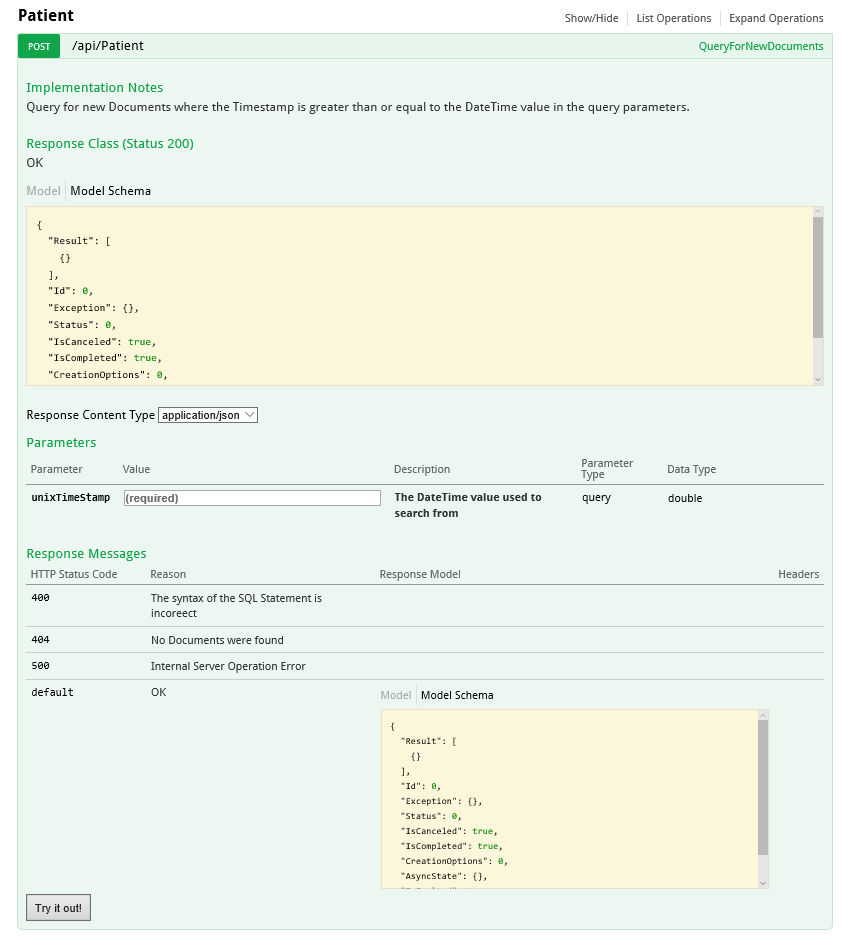
G

#### ConvertToTimeStamp



获取 UTC 日期

#### QueryForNewPatientDocuments



查询

让我们看看此操作背后的代码。

#### GetUtcDate

/// <summary>  
 /// Gets the current UTC Date value  
 /// </summary>  
 /// <returns></returns>  
 [H ttpGet]  
 [Metadata("GetUtcDate", "Gets the current UTC Date value minus the Hours Back")]  
 [SwaggerOperation("GetUtcDate")]  
 [SwaggerResponse(HttpStatusCode.OK, type: typeof (string))]  
 [SwaggerResponse(HttpStatusCode.InternalServerError, "Internal Server Operation Error")]  
 public string GetUtcDate(  
 [Metadata("Hours Back", "How many hours back from the current Date Time")] int hoursBack)  
 {  
  
  
 return DateTime.UtcNow.AddHours(-hoursBack).ToString("r");  
 }

此操作只会返回当前的 UTC 日期时间减去 HoursBack 值。

#### ConvertToTimeStamp

/// <summary>  
 /// Converts DateTime to double  
 /// </summary>  
 /// <param name="currentdateTime"></param>  
 /// <returns></returns>  
 [Metadata("Converts Universal DateTime to number")]  
 [SwaggerResponse(HttpStatusCode.OK, null, typeof (double))]  
 [SwaggerResponse(HttpStatusCode.BadRequest, "DateTime is invalid")]  
 [SwaggerResponse(HttpStatusCode.InternalServerError)]  
 [SwaggerOperation(nameof(ConvertToTimestamp))]  
 public double ConvertToTimestamp(  
 [Metadata("currentdateTime", "DateTime value to convert")] string currentdateTime)  
 {  
 double result;  
  
 try  
 {  
 var uncoded = HttpContext.Current.Server.UrlDecode(currentdateTime);  
  
 var newDateTime = DateTime.Parse(uncoded);  
 //create Timespan by subtracting the value provided from the Unix Epoch  
 var span = newDateTime - new DateTime(1970, 1, 1, 0, 0, 0, 0).ToLocalTime();  
  
 //return the total seconds (which is a UNIX timestamp)  
 result = span.TotalSeconds;  
 }  
 catch (Exception e)  
 {  
 throw new Exception("unable to convert to Timestamp", e.InnerException);  
 }  
  
 return result;  
 }

此操作会将 GetUtcDate 操作的响应转换为双精度值。

#### QueryForNewPatientDocuments

/// <summary>  
 /// Query for new Patient Documents  
 /// </summary>  
 /// <param name="unixTimeStamp"></param>  
 /// <returns>IList</returns>  
 [Metadata("QueryForNewDocuments",  
 "Query for new Documents where the Timestamp is greater than or equal to the DateTime value in the query parameters."  
 )]  
 [SwaggerOperation("QueryForNewDocuments")]  
 [SwaggerResponse(HttpStatusCode.OK, type: typeof (Task<IList<Document>>))]  
 [SwaggerResponse(HttpStatusCode.BadRequest, "The syntax of the SQL Statement is incorrect")]  
 [SwaggerResponse(HttpStatusCode.NotFound, "No Documents were found")]  
 [SwaggerResponse(HttpStatusCode.InternalServerError, "Internal Server Operation Error")]  
 // ReSharper disable once ConsiderUsingAsyncSuffix  
 public IList<Document> QueryForNewPatientDocuments(  
 [Metadata("UnixTimeStamp", "The DateTime value used to search from")] double unixTimeStamp)  
 {  
 var context = new DocumentDbContext();  
 var filterQuery = string.Format(InvariantCulture, "SELECT \* FROM Patient p WHERE p.\_ts >= {0}",  
 unixTimeStamp);  
 var options = new FeedOptions {MaxItemCount = -1};  
  
  
 var collectionLink = UriFactory.CreateDocumentCollectionUri(DocumentDbContext.DatabaseId,  
 DocumentDbContext.CollectionId);  
  
 var response =  
 context.Client.CreateDocumentQuery<Document>(collectionLink, filterQuery, options).AsEnumerable();  
  
 return response.ToList();  
 }

此操作会使用 [DocumentDB .NET SDK](/documentation/articles/documentdb-sdk-dotnet) 创建文档查询。

CreateDocumentQuery<Document>(collectionLink, filterQuery, options).AsEnumerable();

将传入 ConvertToTimeStamp 操作 (unixTimeStamp) 的响应。此操作会返回文档列表 IList<Document>。

我们先前谈到了 CallbackURL。若要在主要逻辑应用中启动工作流，你必须使用 CallbackURL 调用它。

## CallbackURL

若要开始操作，你需要一个 Azure AD 令牌。此令牌可能很难得到。我之前想找到一种简单的方法，Azure 逻辑应用程序管理员 Jeff Hollan 建议在 PowerShell 中使用 [armclient](http://blog.davidebbo.com/2015/01/azure-resource-manager-client.html)。你可以按照所提供的指示进行安装。

你想要使用的操作为“登录”和“调用 ARM API”。

登录：使用相同的凭据登录 Azure 门户。

“调用 ARM API”操作将生成你的 CallBackURL。

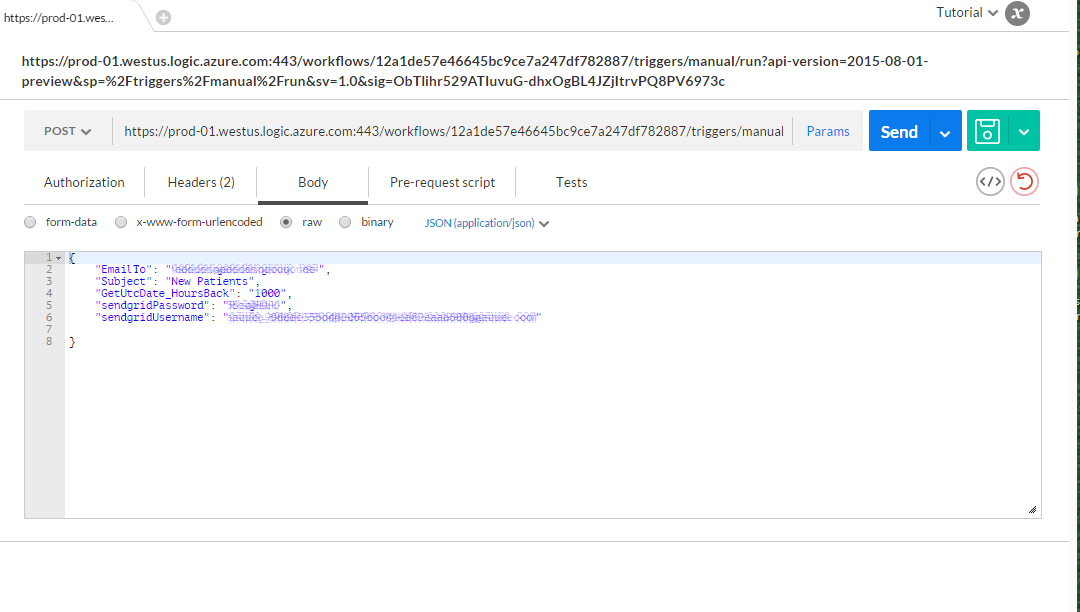
在 PowerShell 中调用它，如下所示：

ArmClient.exe post https://management.azure.com/subscriptions/[YOUR SUBSCRIPTION ID/resourcegroups/[YOUR RESOURCE GROUP]/providers/Microsoft.Logic/workflows/[YOUR LOGIC APP NAME/triggers/manual/listcallbackurl?api-version=2015-08-01-preview

结果应如下所示：

https://prod-02.westus.logic.azure.com:443/workflows/12a1de57e48845bc9ce7a247dfabc887/triggers/manual/run?api-version=2015-08-01-prevaiew&sp=%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=XXXXXXXXXXXXXXXXXXX

你可以使用 [postman](http://www.getpostman.com/) 等工具来测试你的主要逻辑应用，如下图所示。



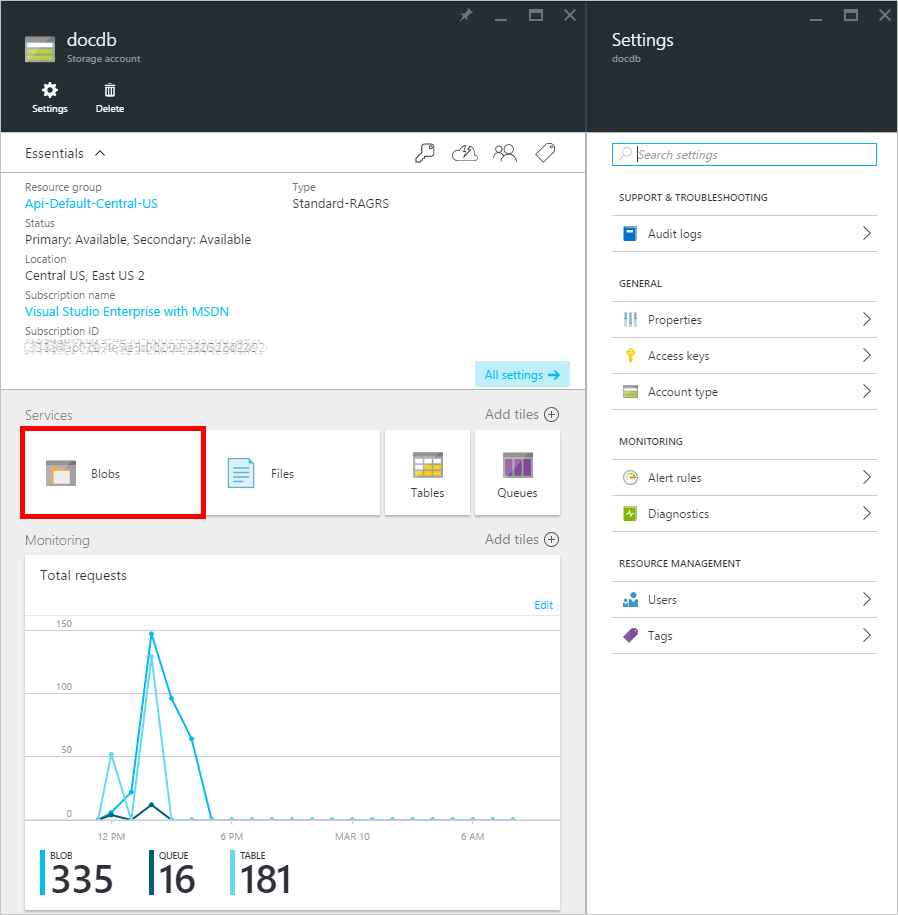
Postman

下表列出的触发器参数构成 DocDB 触发器逻辑应用的主体。

|  |  |
| --- | --- |
| 参数 | 说明 |
| GetUtcDate\_HoursBack | 用于设置搜索开始日期的小时数 |
| sendgridUsername | 用于设置搜索开始日期的小时数 |
| sendgridPassword | Send Grid 电子邮件的用户名 |
| EmailTo | 将会收到电子邮件通知的电子邮件地址 |
| 使用者 | 电子邮件的主题 |

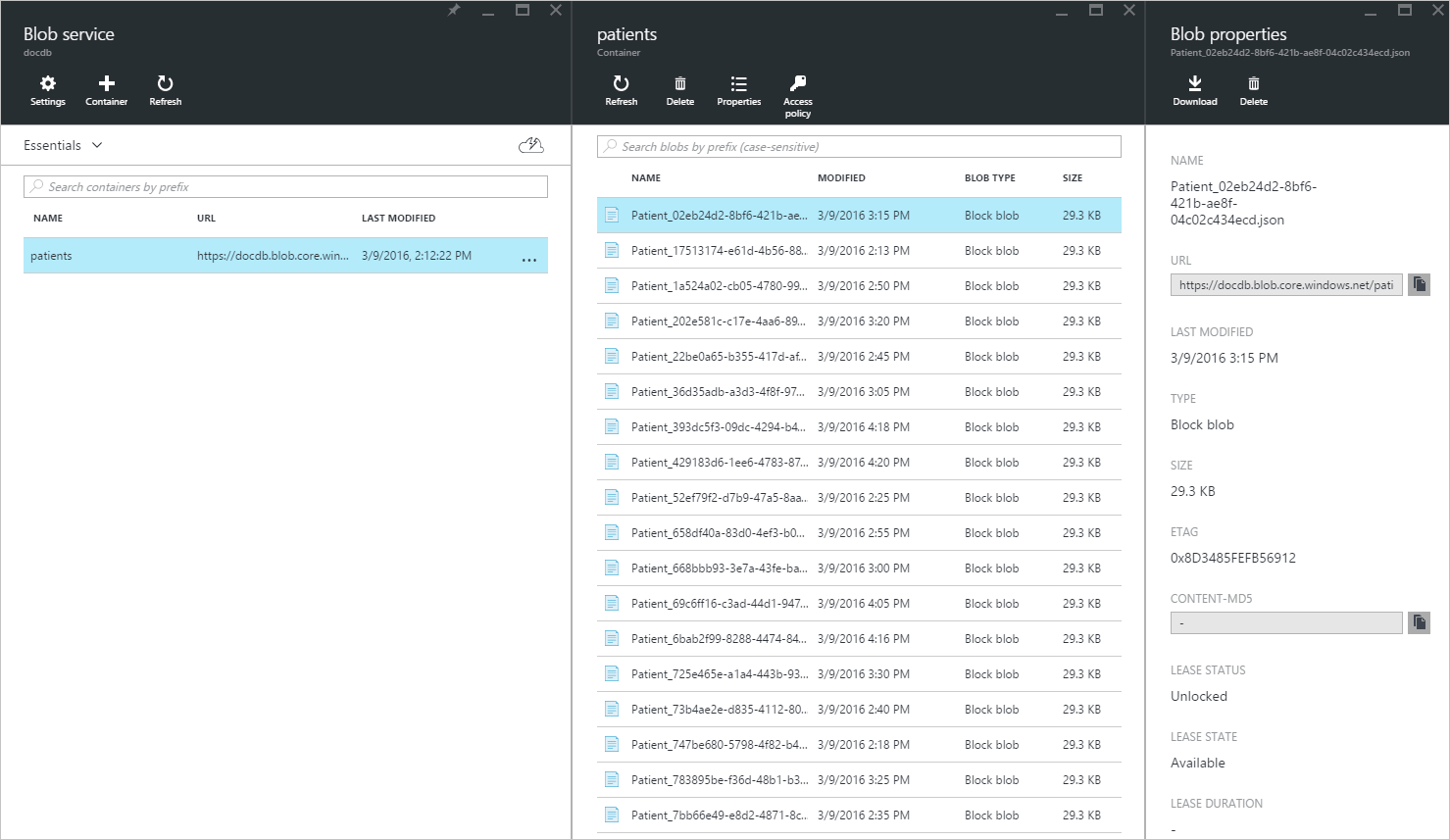
## 在 Azure Blob 服务中查看患者数据

转到 Azure 存储帐户，并选择“服务”下的 Blob，如下图所示。



存储帐户

你将可查看 Patient Blob 文件信息，如下所示。



Blob 服务

## 摘要

在本演练中，你了解了以下信息：

* 可以在 DocumentDB 中实施通知。
* 使用逻辑应用，你可以实现此流程的自动化。
* 使用逻辑应用，你可以减少交付应用程序所需的时间。
* 使用 HTTP，你可以轻松使用逻辑应用内的 API 应用。
* 你可以轻松创建 CallBackURL，以取代 HTTP 侦听程序。
* 你可以利用逻辑应用设计器轻松创建自定义工作流。

重点在于事先规划并建立工作流模型。

## 后续步骤

请下载并使用 [Github](https://github.com/HEDIDIN/DocDbNotifications) 上提供的逻辑应用代码。竭诚邀请你在该应用程序基础上进行构建，并将更改提交到存储库。

若要详细了解 DocumentDB，请访问[学习路径](https://azure.microsoft.com/documentation/learning-paths/documentdb/)。